Project 4 - Hand Tracking / Gesture Recognition

Group 10

Steven Ritchie

Thanh Nguyen

Jeffrey Tang

I. Project Summary

We set out to create an application which could track hand movements and recognize a limited set of gestures (e.g., the sign-language alphabet) in both pre-recorded and live video. We attempted to design and implement our own preprocessing, training, and recognition system, as opposed to relying on pipelines laid out in existing work.

Initially, we experimented with various techniques for hand detection and tracking - most notably contour detection, skin detection, and background subtraction. These techniques proved reasonably effective in controlled settings, but were too brittle for processing live video with diverse backgrounds and lighting conditions.

We then pivoted to focus on an optical flow and machine learning-based approach, whereby we hoped to classify gestures by identifying regions of intense movement.

II. Hand Detection

Our first approach to the problem revolved around using structural cues and prior knowledge of the characteristics (color, skeletal structure, general shape, etc.) of the human hand. We hoped to identify key parts of the hand (specifically, the fingertips and the gaps between them), then use sparse feature tracking to distinguish between gestures.

The first challenge was to create a robust way to determine what region of a given video frame corresponded to a hand. To that end, in the first week of the project, we focused on two preliminary objectives:

1. Create a generic video processing framework to enable rapid iteration on various methods of detection and tracking
2. Implement basic hand-tracking capability in pre-recorded video samples.

This second objective was implemented using a combination of thresholding and edge detection. Given several self-created video samples to work with, we empirically found an acceptable threshold value for separating foreground (the hand) from background. The

thresholded image was then fed to an edge detection algorithm to extract the contour of the hand. Finally, we used OpenCV's built-in capabilities to draw both a convex hull and a tight bounding box around the detected hand contour. We hoped to use the convex hull to identify fingertips and convexity defects, and the bounding box to correct for orientation and relative position on screen.

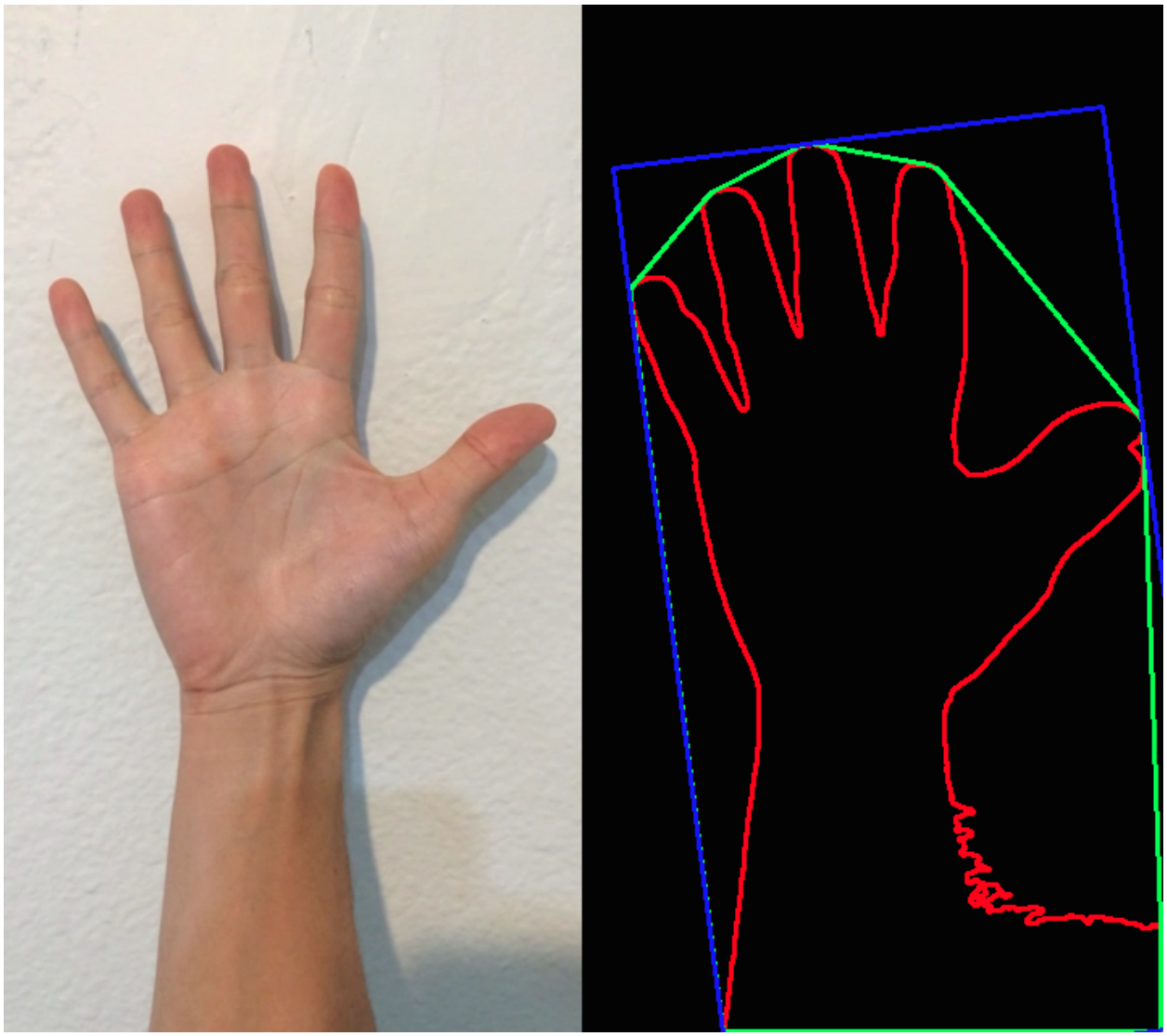


Fig 1. *Left: Original video frame. Right: Processed frame with contour (red), convex hull (green), and bounding box (blue). Even with a very clean background, thresholding can still be confused by shadows, wall textures, etc.*

We quickly discovered that, at least from a practical standpoint, hand detection was far more difficult a problem than the face detection we implemented in Project 3. While robust face

detection could be almost trivially implemented by taking advantage of OpenCV's existing pre-trained Haar cascade classifier, we needed to tune our parameters by hand. This led to problems with robustness; we often missed (or added) parts of the hand due to subtle lighting changes triggered by movement, and different video samples often required different values of the parameters altogether.

III. Optical Flow

The difficulties we encountered in week 1 led us to reconsider our goals. We decided that it would be better to shift our focus towards a different way of recognizing gestures - namely, optical flow. The first and most obvious step in this new direction was to write code to calculate dense optical flow. The initial implementation was fairly straightforward, but we discovered that performance was problematic. Downsampling the input fixed this issue.

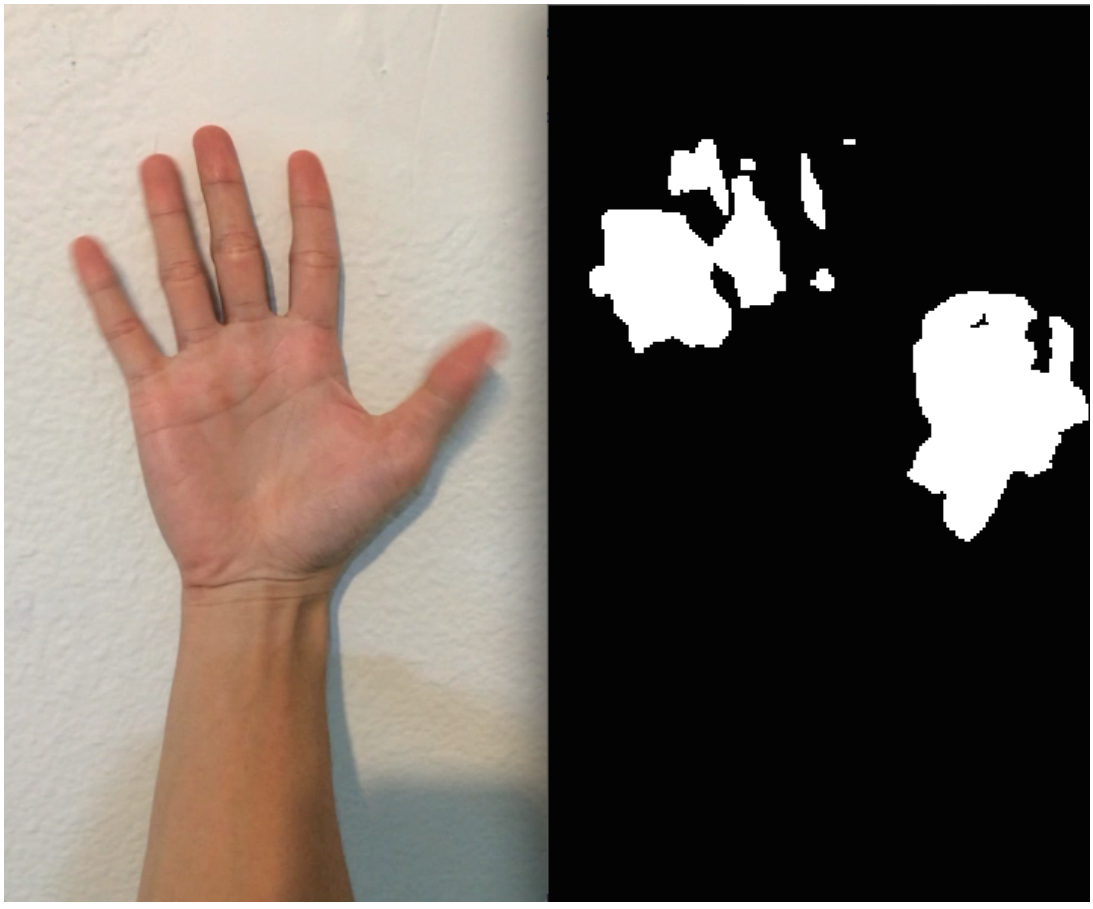


Fig 2. *Left: Original video frame. Right: Downsampled optical flow magnitude.*

Next, since we were primarily interested in regions of movement rather than exact direction of movement, we extracted the magnitudes of the calculated flow vectors and summed them over the length of the video to obtain an energy image - essentially a heat-map describing which areas of the frame saw high amounts of movement.

We also experimented with various ways to refine our hand detection / bounding techniques in order to more robustly handle scale differences in the future. We looked at various combinations of blurring, normalization, thresholding, skin detection, background detection, etc.

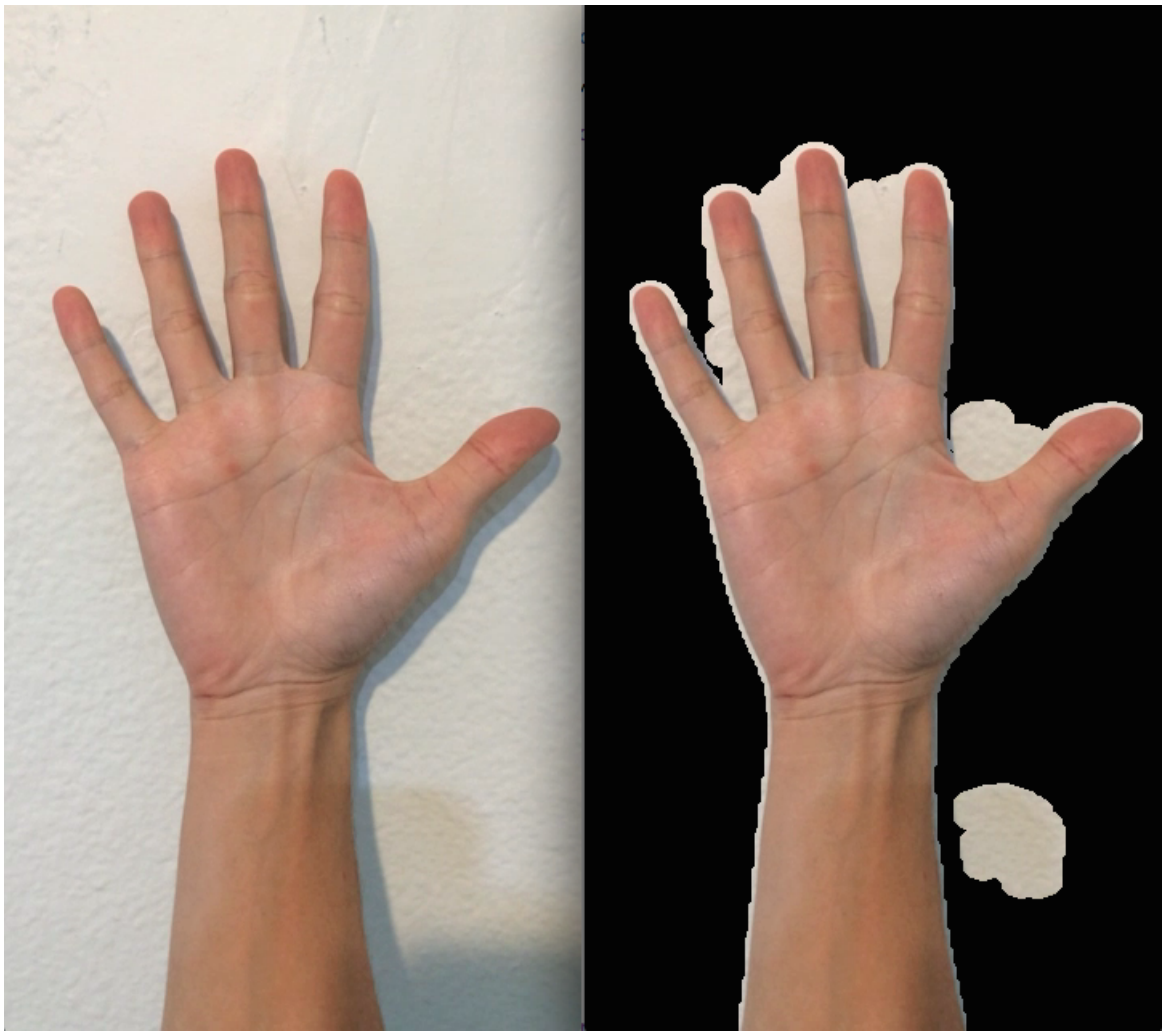


Fig 3. *Left: Original video frame. Right: After applying a mask to isolate skin tones. Can work well given good lighting, but again requires tuning parameters.*

Unfortunately, most of these approaches fell victim to the same pitfalls as the basic thresholding we tried in week 1 - dependence on lighting, clean background, manually tuned parameters.

Finally, to test our code, we also created a small set of training videos covering four sign language gestures - A, B, C, and L. We chose these four because they seemed to involve fairly distinct movements; we hoped that this would make the recognition task easier down the line. At first, recognition was implemented using a naive cost function - given the energy image of an unknown sample to identify, we simply calculated the per-pixel difference between that image and our pre-calculated known images, selecting whichever gesture resulted in the smallest overall difference. This proved overly sensitive to differences in framing.

IV. Classification

Our focus for the final weeks of the project was classification. Because our attempt at a simple cost measure proved ineffective, we decided to fall back onto a machine learning approach using OpenCV's SVM classifier.

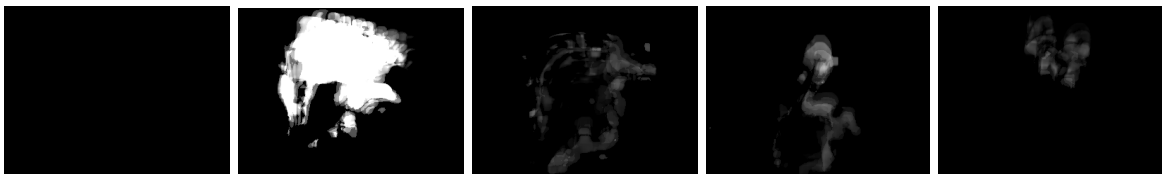
As a proof of concept, we expanded our roster of sample videos to 10 per gesture and fed them into our classifier. The resulting classifier proved able to reliably recall samples on which it had been trained, which suggested that this would be a viable approach.

To that end we created an interactive training/recognition system that allowed capturing of energy images over 20-frame segments of live video. Pressing a number key (e.g., 1, 2, 3) prompts the system to calculate and save an energy image calculated over the next 20 frames of video (or longer, if desired). Pressing ESC ends the collection phase and automatically feeds gathered energy images to a new classifier for training. This was the system demonstrated during our presentation.

Using our training tool, we captured 6 energy images for each of 5 gestures. We used half of the energy images for each gesture to train the SVM and the remaining half to test prediction accuracy. The results were as follows:

predicted: NONE truth: NONE
predicted: NONE truth: NONE
predicted: NONE truth: NONE
predicted: WAVE truth: WAVE
predicted: WAVE truth: WAVE
predicted: WAVE truth: WAVE
predicted: AIRQUOTES truth: AIRQUOTES
predicted: AIRQUOTES truth: AIRQUOTES
predicted: AIRQUOTES truth: AIRQUOTES
predicted: NONE truth: PUNCH
predicted: PUNCH truth: PUNCH
predicted: PUNCH truth: PUNCH
predicted: POINT truth: POINT
predicted: POINT truth: POINT
predicted: AIRQUOTES truth: POINT
Prediction accuracy: 86.6666666667%

After training, we run real-time recognition with labels drawn into the output image. In general, when gestures are performed carefully, the labels are reasonably accurate. However, the labeling does suffer from misclassification when moving between gestures. It is a very noise sensitive system.



none wave punch point airquotes

Fig 4. *Sample energy images for 4 distinct hand gestures*

V. Conclusion

Gesture recognition turned out to be a trickier project than we originally thought. Even the preliminary task of separating a hand from its background turned out to be quite difficult -

especially when trying to adapt to varying conditions. While we were not able to recognize the sign language alphabet as originally planned due to the intricate nature of the symbols, we were able to train an energy image-based classifier that achieved strong results on a set of distinctive gestures, as shown above.